Android Cookbook（影印版）

Android Cookbook

O'REILLY®
東南大學出版社

Ian F. Darwin 编

Android Cookbook（影印版）

Android Cookbook

Ian F. Darwin 编

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

Android Cookbook/（加）达尔文（Darwin, I.F.)编. —影印本. —南京：东南大学出版社，2013.1

书名原文：Android Cookbook

ISBN 978-7-5641-3889-9

I. ① A… II. ①达… III. ①移动终端－应用程序－程序设计 IV. ① TN929.53

中国版本图书馆 CIP 数据核字（2012）第 273594 号

江苏省版权局著作权合同登记

图字：10-2011-407 号

Android Cookbook （影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼 2 号　　邮编：210096
出 版 人：江建中
网　　址：http://www.seupress.com
电子邮件：press@seupress.com
印　　刷：扬中市印刷有限公司
开　　本：787 毫米 × 980 毫米　16 开本
印　　张：43
字　　数：842 千字
版　　次：2013 年 1 月第 1 版
印　　次：2013 年 1 月第 1 次印刷
书　　号：ISBN 978-7-5641-3889-9
定　　价：92.00 元（册）

本社图书若有印装质量问题，请直接与营销部联系。电话（传真）：025-83791830

Table of Contents

Preface xiii

1. Getting Started 1
- 1.1 Introduction: Getting Started 1
- 1.2 Learning the Java Language 1
- 1.3 Creating a "Hello, World" Application from the Command Line 3
- 1.4 Creating a "Hello, World" Application in Eclipse 6
- 1.5 Setting Up an IDE on Windows to Develop for Android 13
- 1.6 Understanding the Android Life Cycle 20
- 1.7 Installing .apk Files onto an Emulator via the ADB 21
- 1.8 Installing Apps onto an Emulator via SlideME 22
- 1.9 Sharing Java Classes from Another Eclipse Project 23
- 1.10 Referencing Libraries to Implement External Functionality 26
- 1.11 Using SDK Samples to Help Avoid Head Scratching 29
- 1.12 Keeping the Android SDK Updated 32
- 1.13 Taking a Screenshot from the Emulator/Android Device 39
- 1.14 Program: A Simple CountDownTimer Example 41
- 1.15 Program: Tipster, a Tip Calculator for the Android OS 44

2. Designing a Successful Application 63
- 2.1 Introduction: Designing a Successful Android Application 63
- 2.2 Exception Handling 66
- 2.3 Accessing Android's Application Object as a "Singleton" 69
- 2.4 Keeping Data When the User Rotates the Device 71
- 2.5 Monitoring the Battery Level of an Android Device 74
- 2.6 Creating Splash Screens in Android 75
- 2.7 Designing a Conference/Camp/Hackathon/Institution App 79
- 2.8 Using Google Analytics in an Android Application 81
- 2.9 A Simple Torch/Flashlight 83
- 2.10 Adapting an Android Phone Application to Be Used on a Tablet 86
- 2.11 Setting First-Run Preferences 88

2.12 Formatting the Time and Date for Display 89
2.13 Controlling Input with KeyListeners 91
2.14 Backing Up Android Application Data 95
2.15 Using Hints Instead of Tool Tips 101

3. Testing 103
3.1 Introduction: Testing 103
3.2 Doing Test-Driven Development (TDD) in Android 103
3.3 Setting Up an Android Virtual Device (AVD) for App Testing 104
3.4 Testing on a Huge Range of Devices with Cloud-based Testing 113
3.5 Creating and Using a Test Project 114
3.6 Troubleshooting Application Crashes 118
3.7 Debugging Using Log.d and LogCat 122
3.8 Getting Bug Reports from Users Automatically with BugSense 123
3.9 Using a Local Runtime Application Log for Analysis of Field Errors or Situations 125
3.10 Reproducing Activity Life-Cycle Scenarios for Testing 129
3.11 Keeping Your App Snappy with StrictMode 134
3.12 Running the Monkey Program 135
3.13 Sending Text Messages and Placing Calls Between AVDs 137

4. Inter-/Intra-Process Communication 141
4.1 Introduction: Inter-/Intra-Process Communication 141
4.2 Opening a Web Page, Phone Number, or Anything Else with an Intent 142
4.3 Emailing Text from a View 143
4.4 Sending an Email with Attachments 146
4.5 Pushing String Values Using Intent.putExtra() 147
4.6 Retrieving Data from a Subactivity Back to Your Main Activity 149
4.7 Keeping a Service Running While Other Apps Are on Display 151
4.8 Sending/Receiving a Broadcast Message 153
4.9 Starting a Service After Device Reboot 154
4.10 Creating a Responsive Application Using Threads 155
4.11 Using AsyncTask to Do Background Processing 157
4.12 Sending Messages Between Threads Using an Activity Thread Queue and Handler 165
4.13 Creating an Android Epoch HTML/JavaScript Calendar 167

5. Content Providers 173
5.1 Introduction: Content Providers 173
5.2 Retrieving Data from a Content Provider 173
5.3 Writing a Content Provider 175
5.4 Writing an Android Remote Service 177

6. Graphics 183
6.1 Introduction: Graphics 183
6.2 Using a Custom Font 183
6.3 Drawing a Spinning Cube with OpenGL ES 186
6.4 Adding Controls to the OpenGL Spinning Cube 190
6.5 Freehand Drawing Smooth Curves 193
6.6 Taking a Picture Using an Intent 199
6.7 Taking a Picture Using android.media.Camera 201
6.8 Scanning a Barcode or QR Code with the Google ZXing Barcode Scanner 204
6.9 Using AndroidPlot to Display Charts and Graphs 208
6.10 Using Inkscape to Create an Android Launcher Icon 210
6.11 Creating Easy Launcher Icons from OpenClipArt.org Using Paint.NET 217
6.12 Using Nine Patch Files 224
6.13 Creating HTML5 Charts with Android RGraph 227
6.14 Adding a Simple Raster Animation 231
6.15 Using Pinch to Zoom 234

7. Graphical User Interface 239
7.1 Introduction: GUI 239
7.2 Understanding and Following User Interface Guidelines 240
7.3 Handling Configuration Changes by Decoupling the View from the Model 242
7.4 Creating a Button and Its Click Event Listener 245
7.5 Wiring Up an Event Listener in Five Different Ways 246
7.6 Using CheckBoxes and RadioButtons 250
7.7 Enhancing UI Design Using Image Buttons 254
7.8 Offering a Drop-Down Chooser via the Spinner Class 256
7.9 Handling Long-Press/Long-Click Events 258
7.10 Displaying Text Fields with TextView and EditText 259
7.11 Constraining EditText Values with Attributes and the TextWatcher Interface 260
7.12 Implementing AutoCompleteTextView 263
7.13 Feeding AutoCompleteTextView Using an SQLite Database Query 265
7.14 Turning Edit Fields into Password Fields 267
7.15 Changing the Enter Key to "Next" on the Soft Keyboard 268
7.16 Processing Key-Press Events in an Activity 270
7.17 Let Them See Stars: Using RatingBar 272
7.18 Making a View Shake 276
7.19 Providing Haptic Feedback 277
7.20 Navigating Different Activities Within a TabView 281
7.21 Creating a Custom Title Bar 283

7.22 Formatting Numbers 285
7.23 Formatting with Correct Plurals 289
7.24 Starting a Second Screen from the First 292
7.25 Creating a Loading Screen That Will Appear Between Two Activities 301
7.26 Using SlidingDrawer to Overlap Other Components 303
7.27 Customizing the SlidingDrawer Component to Animate/Transition from the Top Down 305
7.28 Adding a Border with Rounded Corners to a Layout 307
7.29 Detecting Gestures in Android 309
7.30 Building a UI Using Android 3.0 Fragments in Android 1.6 and Later 316
7.31 Using the Android 3.0 Photo Gallery 321
7.32 Creating a Simple App Widget 324

8. GUI Alerts: Menus, Dialogs, Toasts, and Notifications 329
8.1 Introduction: GUI Alerts 329
8.2 Creating and Displaying a Menu 330
8.3 Handling Choice Selection in a Menu 331
8.4 Creating a Submenu 333
8.5 Creating a Pop-up/Alert Dialog 336
8.6 Using a Timepicker Widget 338
8.7 Creating an iPhone-like Wheel Picker for Selection 340
8.8 Creating a Tabbed Dialog 343
8.9 Creating a ProgressDialog 346
8.10 Creating a Custom Dialog with Buttons, Images, and Text 347
8.11 Creating a Reusable About Box Class 349
8.12 Customizing the Appearance of a Toast 353
8.13 Creating a Notification in the Status Bar 354

9. GUI: ListView 361
9.1 Introduction: ListView 361
9.2 Building List-Based Applications with ListView 361
9.3 Creating a "No Data" View for ListViews 366
9.4 Creating an Advanced ListView with Images and Text 367
9.5 Using Section Headers in ListViews 372
9.6 Keeping the ListView with the User's Focus 377
9.7 Writing a Custom List Adapter 377
9.8 Handling Orientation Changes: From ListView Data Values to Landscape Charting 381

10. Multimedia 387
10.1 Introduction: Multimedia 387

10.2 Playing a YouTube Video 387
10.3 Using the Gallery with the ImageSwitcher View 388
10.4 Capturing Video Using MediaRecorder 391
10.5 Using Android's Face Detection Capability 394
10.6 Playing Audio from a File 398
10.7 Playing Audio Without Interaction 400
10.8 Using Speech to Text 402
10.9 Making the Device Speak with Text-to-Speech 403

11. Data Persistence 407
11.1 Introduction: Data Persistence 407
11.2 Getting File Information 407
11.3 Reading a File Shipped with the App Rather Than in the Filesystem 411
11.4 Listing a Directory 413
11.5 Getting Total and Free Space Information About the SD Card 414
11.6 Providing User Preference Activity with Minimal Effort 415
11.7 Checking the Consistency of Default Shared Preferences 419
11.8 Performing Advanced Text Searches 421
11.9 Creating an SQLite Database in an Android Application 427
11.10 Inserting Values into an SQLite Database 428
11.11 Loading Values from an Existing SQLite Database 428
11.12 Working with Dates in SQLite 429
11.13 Parsing JSON Using JSONObject 432
11.14 Parsing an XML Document Using the DOM API 433
11.15 Parsing an XML Document Using an XmlPullParser 435
11.16 Adding a Contact 439
11.17 Reading Contact Data 442

12. Telephone Applications 445
12.1 Introduction: Telephone Applications 445
12.2 Doing Something When the Phone Rings 445
12.3 Processing Outgoing Phone Calls 449
12.4 Dialing the Phone 453
12.5 Sending Single-Part or Multipart SMS Messages 454
12.6 Receiving an SMS Message in an Android Application 457
12.7 Using Emulator Controls to Send SMS Messages to the Emulator 458
12.8 Using Android's TelephonyManager to Obtain Device Information 459

13. Networked Applications 471
13.1 Introduction: Networking 471
13.2 Using a RESTful Web Service 472
13.3 Extracting Information from Unstructured Text Using Regular Expressions 474

13.4 Parsing RSS/Atom Feeds Using ROME 476
13.5 Using MD5 to Digest Clear Text 481
13.6 Converting Text into Hyperlinks 481
13.7 Accessing a Web Page Using WebView 482
13.8 Customizing a WebView 484

14. Gaming and Animation 485
14.1 Introduction: Gaming and Animation 485
14.2 Building an Android Game Using flixel-android 486
14.3 Building an Android Game Using AndEngine (Android-Engine) 489
14.4 Processing Timed Keyboard Input 495

15. Social Networking 497
15.1 Introduction: Social Networking 497
15.2 Integrating Social Networking Using HTTP 497
15.3 Loading a User's Twitter Timeline Using JSON 500

16. Location and Map Applications 503
16.1 Introduction: Location-Aware Applications 503
16.2 Getting Location Information 503
16.3 Accessing GPS Information in Your Application 505
16.4 Mocking GPS Coordinates on a Device 508
16.5 Using Geocoding and Reverse Geocoding 510
16.6 Getting Ready for Google Maps Development 511
16.7 Adding a Device's Current Location to Google Maps 517
16.8 Drawing a Location Marker on a Google MapView 519
16.9 Drawing Multiple Location Markers on a MapView 523
16.10 Creating Overlays for a Google MapView 528
16.11 Changing Modes of a Google MapView 529
16.12 Drawing an Overlay Icon Without Using a Drawable 530
16.13 Implementing Location Search on Google Maps 535
16.14 Placing a MapView Inside a TabView 537
16.15 Handling a Long-Press in a MapView 541
16.16 Using OpenStreetMap 544
16.17 Creating Overlays in OpenStreetMap Maps 547
16.18 Using a Scale on an OpenStreetMap Map 550
16.19 Handling Touch Events on an OpenStreetMap Overlay 551
16.20 Getting Location Updates with OpenStreetMap Maps 554

17. Accelerometer 559
17.1 Introduction: Sensors 559
17.2 Checking for the Presence or Absence of a Sensor 560
17.3 Using the Accelerometer to Detect Shaking of the Device 561

17.4 Checking Whether a Device Is Facing Up or Facing Down Based on Screen Orientation Using an Accelerometer 564
17.5 Finding the Orientation of an Android Device Using an Orientation Sensor 565
17.6 Reading the Temperature Sensor 567

18. Bluetooth 569
18.1 Introduction: Bluetooth 569
18.2 Enabling Bluetooth and Making the Device Discoverable 569
18.3 Connecting to a Bluetooth-Enabled Device 571
18.4 Listening for and Accepting Bluetooth Connection Requests 574
18.5 Implementing Bluetooth Device Discovery 575

19. System and Device Control 577
19.1 Introduction: System and Device Control 577
19.2 Accessing Phone Network/Connectivity Information 577
19.3 Obtaining Information from the Manifest File 578
19.4 Changing Incoming Call Notification to Silent, Vibrate, or Normal 579
19.5 Copying Text and Getting Text from the Clipboard 581
19.6 Using LED-Based Notifications 582
19.7 Making the Device Vibrate 583
19.8 Running Shell Commands from Your Application 584
19.9 Determining Whether a Given Application Is Running 586

20. Other Programming Languages and Frameworks 587
20.1 Introduction: Other Programming Languages 587
20.2 Running an External/Native Unix/Linux Command 588
20.3 Running Native C/C++ Code with JNI on the NDK 589
20.4 Getting Started with the Scripting Layer for Android (SL4A, Formerly Android Scripting Environment) 594
20.5 Creating Alerts in SL4A 597
20.6 Fetching Your Google Documents and Displaying Them in a ListView Using SL4A 600
20.7 Sharing SL4A Scripts in QR Codes 603
20.8 Using Native Handset Functionality from WebView via JavaScript 607
20.9 Creating a Platform-Independent Application Using PhoneGap/Cordova 608

21. Strings and Internationalization 611
21.1 Introduction: Internationalization 611
21.2 Internationalizing Application Text 612
21.3 Finding and Translating Strings 615

21.4 Handling the Nuances of strings.xml 617

22. Packaging, Deploying, and Distributing/Selling Your App 623
22.1 Introduction: Packaging, Deploying, and Distributing 623
22.2 Creating a Signing Certificate 623
22.3 Signing Your Application 626
22.4 Distributing Your Application via Android Play (formerly the Android Market) 627
22.5 Integrating AdMob into Your App 629
22.6 Obfuscating and Optimizing with ProGuard 633
22.7 Providing a Link to Other Published Apps in the Google Play Market 636

Index 641

CHAPTER 1

Getting Started

1.1 Introduction: Getting Started

Ian Darwin

Discussion

The famous "Hello, World" pattern came about when Kernighan and Plaugher wanted to write a "recipe" on how to get started in any new programming language and environment. This chapter is affectionately dedicated to these fine gentlemen, and to everyone who has ever struggled to get started in a new programming paradigm.

1.2 Learning the Java Language

Ian Darwin

Problem

Android apps are written in the Java programming language before they are converted into Android's own class file format, DEX. If you don't know how to program in Java you will find it hard to write Android apps.

Solution

Lots of resources are available for learning Java. Most of them will teach you what you need, but will also mention some API classes that are not available for Android development. *Avoid* any sections in any resource that talk about topics listed in the lefthand column of Table 1-1.

Table 1-1. Parts of the Java API to ignore

Java API	Android equivalent
Swing, applets	Android's GUI; see Chapter 7.
Application entry point `main()`	See Recipe 1.6.
J2ME/Java ME	Most of `android.*` replaces Java ME API.
Servlets/JSP, J2EE/Java EE	Designed for server-side use.

Discussion

Here are some books and resources on Java programming:

- *Java in a Nutshell* (*http://shop.oreilly.com/product/9780596007737.do*) by David Flanagan (O'Reilly) is a good introduction for programmers, particularly those who are coming from C/C++. This book has grown from an acorn to a coconut in size, to keep up with the growth of Java SE over its lifetime.
- *Head First Java* (*http://shop.oreilly.com/product/9780596009205.do*) by Kathy Sierra and Bert Bates (O'Reilly). This provides a great visual-learner-oriented introduction to the language.
- *Thinking in Java* (*http://www.mindview.net/Books/TIJ4*) by Bruce Eckel (Prentice-Hall).
- *Learning Java* (*http://shop.oreilly.com/product/9780596008734.do*) by Patrick Niemeyer and Jonathan Knudsen (O'Reilly).
- "Great Java: Level 1" (*http://shop.oreilly.com/product/9780596809393.do*), a video by Brett McLaughlin (O'Reilly). This provides a visual introduction to the language.
- *Java: The Good Parts* (*http://shop.oreilly.com/product/9780596803742.do*) by Jim Waldo (O'Reilly).
- *Java Cookbook* (*http://shop.oreilly.com/product/9780596007010.do*), which I wrote and which O'Reilly published. This is regarded as a good second book for Java developers. It has entire chapters on strings, regular expressions, numbers, dates and time, structuring data, I/O and directories, internationalization, threading, and networking, all of which apply to Android. It also has a number of chapters that are specific to Swing and to some EE-based technologies.

Please understand that this list will probably never be completely up-to-date. You should also refer to O'Reilly's freely downloadable (with registration) *Android Development Bibliography* (*http://shop.oreilly.com/product/0636920021896.do*), a compilation of all the books from the various publishers whose books are in the online Safari service. This book is also distributed without charge at relevant conferences where O'Reilly has a booth.

See Also

This book's primary author maintains a list of Java resources online at *http://www.darwinsys.com/java/*.

O'Reilly has many of the best Java books around; there's a complete list at *http://oreilly.com/pub/topic/java*.

1.3 Creating a "Hello, World" Application from the Command Line

Ian Darwin

Problem

You want to create a new Android project without using the Eclipse ADT plug-in.

Solution

Use the Android Development Kit (ADK) tool `android` with the `create project` argument and some additional arguments to configure your project.

Discussion

In addition to being the name of the platform, `android` is also the name of a command-line tool for creating, updating, and managing projects. You can either navigate into the *android-sdk-xxx* directory, or you can set your `PATH` variable to include its *tools* subdirectory.

Then, to create a new project, give the command `android create project` with some arguments. Example 1-1 is an example run under MS-DOS.

Example 1-1. Creating a new project

```
C:> PATH=%PATH%;"C:\Documents and Settings\Ian\My Documents\android-sdk-windows\tools"; \
          "C:\Documents and Settings\Ian\My Documents\android-sdk-windows\platform-tools"
C:> android create project --target android-7 --package com.example.foo
     --name Foo --activity FooActivity --path .\MyAndroid
Created project directory: C:\Documents and Settings\Ian\My Documents\MyAndroid
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\src\com\example\foo
Added file C:\Documents and Settings\Ian\My
    Documents\MyAndroid\src\com\example\foo\FooActivity.java
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\bin
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\libs
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res\values
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\res\values\strings.xml
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res\layout
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\res\layout\main.xml
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\AndroidManifest.xml
```

```
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\build.xml

C:>
```

Table 1-2 lists the arguments for the `create project` code.

Table 1-2. List of create project arguments

Name	Meaning	Example
`--activity`	Name of your "main class" and default name for the generated *.apk* file.	`--activity HelloActivity`
`--name`	Name of the project and the generated *.apk* file.	`--name MyProject`
`--package`	Name of the Java package for your classes.	`--package com.example.hello`
`--path`	Path to create the project in (does not create a subdirectory under this, so don't use `/home/you/workspace`, but rather `/home/you/workspace/NewProjectName`).	`--path /home/ian/workspace/MyProject` (see above for Windows example)
`--target`	API level of the Android platform to target; use `android list targets` to see list of targets. A number is an "ID," not an API level; for that, use `android-` with the API level you want.	`--target android-10`

If it cannot complete the requested operation, the `android` command presents a voluminous "command usage" message listing all the operations it can do and the arguments for them. If successful, the `android create project` command creates the files and directories listed in Table 1-3.

Table 1-3. Artifacts created by create project

Name	Meaning
AndroidManifest.xml	Config file that tells Android about your project
bin	Generated binaries (compiled class files)
build.properties	Editable properties file
build.xml	Standard Ant build control file
default.properties or *project.properties* (depending on tools version)	Stores SDK version and libraries used; maintained by plug-in
gen	Generated stuff
libs	Libraries, of course
res	Important resource files (*strings.xml*, layouts, etc.)
src	Source code for your application
src/packagename/ActivityName.java	Source of "main" starting activity
test	Copies of most of the above

It is a normal and recommended Android practice to create your user interface in XML using the layout file created under `res/layout`, but it is certainly possible to write all the code in Java. To keep this example self-contained, we'll do it the "wrong" way for now. Use your favorite text editor to replace the contents of the file *HelloWorld.java* with the contents of Example 1-2.

Example 1-2. HelloWorld.java

```
import android.app.Activity;
import android.widget.*;

public class Hello extends Activity {

        /**
         * This method gets invoked when the activity is instantiated in
         * response to e.g., you clicked on the app's Icon in the Home Screen.
         */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                // Create a TextView for the current Activity
                TextView view = new TextView(this);
                // Make it say something
                view.setText("Hello World");
                // Put this newly created view into the Activity,
                // sort of like JFrame.getContentPane().add(view)
                setContentView(view);
        }
}
```

Assuming you have the Apache Software Foundation Ant Build Tool (*http://ant.apache.org/*) installed (and it is included with recent versions of the Android SDK), you can now (in a command-line window) change to the project directory (`...MyDocuments\MyAndroid` in Example 1-1) and issue the command:

```
ant debug
```

This will create an archive file named, for example, *MyAndroid.apk* (with "apk" standing for Android Package) in the ***bin*** directory.

If this is your first time here, you may need to create an Android Virtual Device (AVD), which is just a named configuration for the Android emulator specifying target resolution, API level, and so on. You can create an emulator using:

```
android create avd -n my_droid -t 7
```

For more details on creating an AVD, see Recipe 3.3.

You can then start the Android Debug Bridge (ADB) server and the emulator:

```
adb start-server
emulator -avd my_droid -t 5
```

Assuming you now have either the emulator running or your device plugged in and recognized via USB, you can then do:

```
adb -e install -r bin/MyAndroid.apk
```

The `-e` flag is for the emulator; use `-d` for a real device.

If you are handy with shell scripts or batch files, you'll want to create one called, say, *download*, to avoid typing the `adb` invocation on every build cycle.

Finally you can start your app! You can use the Application list: tap the little icon that looks like a 5×5 row of dots, scroll to your application by name, and tap its icon.

You will probably find it convenient to create an icon for your app on the home screen of the device or emulator; this icon will survive multiple `install -r` cycles, so it's the easiest way to test the running of your application.

See Also

Recipe 1.4. The blog "a little madness" has a more detailed formulation (*http://www.alittlemadness.com/2010/05/31/setting-up-an-android-project-build/*). The official Android reference site has a page on developing without Eclipse (*http://developer.android.com/guide/developing/other-ide.html*).

1.4 Creating a "Hello, World" Application in Eclipse

Ian Darwin

Problem

You want to use Eclipse to develop your Android application.

Solution

Install Eclipse (*http://www.eclipse.org/*), the Android SDK (*http://developer.android.com/sdk/*), and the ADT plug-in (*http://developer.android.com/sdk/eclipse-adt.html*). Create your project and start writing your app. Build it, and test it under the emulator, from within Eclipse.

Discussion

Once you have these items installed, you are ready to begin:

- Eclipse IDE (*http://www.eclipse.org/*)
- The Android SDK (*http://developer.android.com/sdk/*)
- The ADT plug-in (*http://developer.android.com/sdk/eclipse-adt.html*)

If you want a more detailed exposition of installing these three items, please refer to Recipe 1.5.